**NAME : MANGESH A. GHADWAJE**

**ROLL NO:24**

**BATCH : B2**

**COURSE: ML PRACTICAL**

**Assginment No. 5**

**Problem Statement :**

**Design and implement SVM for classification with the data set**

**given in the assignment No. 4. Test for Accuracy and Precision.**

**Also Analyze the performance of SVM and KNN**

**Code :**

***import matplotlib.pyplot as plt***

***import pandas as pd***

***import numpy as np***

***from sklearn.preprocessing import StandardScaler***

***from sklearn.svm import SVC***

***from sklearn.model\_selection import train\_test\_split***

***from sklearn.datasets import make\_classification***

***from sklearn.metrics import classification\_report, accuracy\_score***

***# Load dataset***

***file\_path = 'emails.csv'***

***df = pd.read\_csv(file\_path)***

***# Generate synthetic data for classification (as an example)***

***X, y = make\_classification(n\_samples=100, n\_features=2, n\_classes=2, n\_redundant=0, n\_clusters\_per\_class=1)***

***# Split the dataset into training and testing sets***

***X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=90)***

***# Standardize the features***

***sc = StandardScaler()***

***X\_train = sc.fit\_transform(X\_train)***

***X\_test = sc.transform(X\_test)***

***# Create and train the SVM model***

***svm = SVC(kernel='rbf', random\_state=0)***

***svm.fit(X\_train, Y\_train)***

***# Predict on the test set***

***y\_pred = svm.predict(X\_test)***

***# Calculate accuracy***

***accuracy = accuracy\_score(Y\_test, y\_pred)***

***# Print results***

***print("Standardized Training Data:\n", X\_train)***

***print("\nStandardized Testing Data:\n", X\_test)***

***print("\nClassification Report:\n", classification\_report(Y\_test, y\_pred))***

***print(f"Accuracy: {accuracy:.2f}")***

***# Function to plot decision boundaries***

***def plot\_decision\_boundaries(X, y, model, ax):***

***h = .02 # step size in the mesh***

***x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1***

***y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1***

***xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))***

***Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])***

***Z = Z.reshape(xx.shape)***

***ax.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)***

***scatter = ax.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', marker='o', cmap=plt.cm.coolwarm)***

***legend1 = ax.legend(\*scatter.legend\_elements(), title="Classes")***

***ax.add\_artist(legend1)***

***ax.set\_xlim(xx.min(), xx.max())***

***ax.set\_ylim(yy.min(), yy.max())***

***ax.set\_xticks(())***

***ax.set\_yticks(())***

***# Plot decision boundaries***

***fig, ax = plt.subplots()***

***plot\_decision\_boundaries(X\_test, Y\_test, svm, ax)***

***plt.title('SVM Decision Boundary')***

***plt.show()***

**Output :**

